JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

TEST -3 EXAMINATION- May 2017

B.Tech/M.Tech VIII/II Semester

COURSE CODE: 10M11CI211 MAX. MARKS:35

COURSE NAME: Advanced Algorithm

COURSE CREDITS: 3 MAX. TIME: 2 Hrs

Note: All questions are compulsory. Carrying of mobile phone during examinations will be treated as case of unfair means.

Question 1:

1. Find the Shapley value of the game with characteristic function **[6 Marks]**

$$v(\emptyset) = 0 \quad \begin{array}{ll} v(\{1\}) = 1 & v(\{1,2\}) = 2 \\ v(\{2\}) = 0 & v(\{1,3\}) = -1 \\ v(\{3\}) = -4 & v(\{2,3\}) = 3 \end{array} \quad v(\{1,2,3\}) = 6$$

2. Find the characteristic function of the 3-person game with players I, II, and III with two pure strategies each and with the following payoff vectors. Note that this is a zero-sum game. Hence, v({1, 3}) = −v({2}), etc. **[4 Marks]**

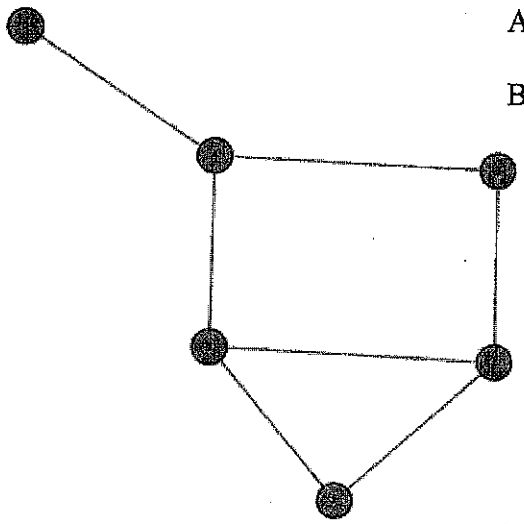
If I chooses 1:

II: \ III:	1	2
1	(−2, 1, 1)	(1, −4, 3)
2	(1, 3, −4)	(10, −5, −5)

If I chooses 2

II: \ III:	1	2
1	(−1, −2, 3)	(−4, 2, 2)
2	(12, −6, −6)	(−1, 3, −2)

Question 2: Consider a graph:



A. Compute: Degree Centrality, Closeness Centrality and Betweenness centrality **[6 Marks]**

B. Explain connected dominating set? What is the connected dominating set of the given figure? **[4 Marks]**

Question 3:

1. Find the complexity of given code **[2 Marks]**

```
// Here c is a constant greater than 1
for (int i = 2; i <=n; i = pow(i, c)) {
    // some O(1) expressions
}
//Here fun is sqrt or cuberoot or any other constant root
for (int i = n; i > 0; i = fun(i)) {
    // some O(1) expressions
}
```

2. Let S be an NP-complete problem and Q and R be two other problems not known to be in NP. Q is polynomial time reducible to S and S is polynomial-time reducible to R. Then, we can say R is NP-hard, justify the answer with proper reasoning? **[2 Marks]**

3. What is chromatic number, explain it with the help of example ? **[2 Marks]**

4. Use Masters Theorem: **[3 Marks]**
   a. $T(n) = 2^n T(n/2) + n^n$
   b. $T(n) = 16T(n/4) + n$

5. Given a number "n", write an algorithm using dynamic programming to find the least number of perfect square numbers sum needed to get "n". Also write your algorithm's complexity **[6 Marks]**

Example:

n=12, return 3 (4 + 4 + 4) = (2^2 + 2^2 + 2^2) NOT (3^2 + 1 + 1 + 1)
Similarly
n = 6, return 3 (4 + 1 + 1) = (2^2 + 1^2 + 1^2)